IOS XR on Linux

Bring-up plan

Modification History

Revision	Date	Originator	Comments
0.01	8 May 2008	Etienne Martineau	Initial Draft
0.02	21 May 2008	Etienne Martineau	PCI & network
0.03	18 June 2008	Etienne Martineau	Harddrive & File system

Table of contents

Table of Contents

1 Linux BSP for Cisco IOS-XR hfr-ppc 7455 Blade....3

1.1 ROMMON environment setting. DONE....3
1.2 ROMMON elf header limitation. WORKAROUND....3
1.3 ROMMON elf section limitation. DONE....3
1.4 L1 cache invalidate hangs the CPU. WORKAROUND....4
1.5 Marvell Bridge initialization hangs. DONE....4
1.6 External Serial Port adjustment and Interruption Mapping. DONE....5
1.7 Watchdog deactivation. DONE....5
1.8 Get Memory size broken for 4GB of RAM. DONE....5
1.9 Networking. DONE....6
1.9.1 PHY address adjustment. DONE....6
1.9.2 Special PHY setting for hfr blade. DONE....6
1.9.3 System hangs when configuring network. DONE....6
1.9.4 Marvell Rx/Tx DMA descriptor not working correctly. WORKAROUND....6
1.9.5 Only 1 port active. DONE....7
1.9.6 Active -> standby limited connectivity. DONE....7
1.10 PCI. DONE....8
1.11 Build environment. DONE....9
1.12 Disk Driver and Filesystem. DONE....9

2 Works that need to be done....9

2.1 Kernel support for 4Gb of RAM. TODO....9
2.2 Kernel version. TODO....10
2.3 Root file system. TODO....10

3 Annex....11

3.1 Linux booting console output....11
3.2 Linux Kernel object section....13
3.3 Linux wrapper object section....15

1 Linux BSP for Cisco IOS-XR hfr-ppc 7455 Blade

Linux is now booting on the hfr-ppc 7455 Blade! See the console output

1.1 ROMMON environment setting. DONE

Under Rommon the image checksum MUST be disabled “TFTP_CHECKSUM=0”otherwise the system will hang. This is basically because Linux image does not use the same wrapper as comp-hfr-mini.vm and thus confuse Rommon when computing the checksum.

1.2 ROMMON elf header limitation. WORKAROUND

When loading a Linux image Rommon return the following error:

```
tftp_process_packet: last packet, block=5617, size=499.File reception completed.
e_phnum 2
TFTP flash copy: Invalid file header, copy terminated.
 Reason: Wrong target platform OR corrupt file.
```

Rommon code inspection reveals that e_phnum has to be set to 1. The Linux image has e_phnum set to 2.

The **workaround** consist of using a Hex viewer to poke the correct value into the elf header section.

The permanent solution would be to either fix Rommon OR provide an automated tools to fix the elf header of Linux image.

1.3 ROMMON elf section limitation. DONE

When loading a Linux image Rommon return the following error:

```
Receiving /wsetmartin/vmlinux from 223.255.254.254 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
tftp_process_packet: last packet, block=5617, size=499.File reception completed.
Headersize=2875891, oursize=2875891 loadprog: error - (ELF) too many loadable sections
```

Rommon has support for only 8 (#define LOADTBLENTRIES 8) section where Linux has 35 (see Linux kernel section).

It's has been decided to use a wrapper on top of the Linux kernel. The standard PPC 6xx wrapper has been used with minor modification for External Uart and Marvell Bridge. The wrapper has only 4 section so it's compatible with Rommon (see Linux wrapper object section)

The wrapper does the following:

- Begin at some arbitrary location in RAM or Flash. Rommon launch the Image @0x0080_0000.
- Initialize core registers
- Move the boot code to the link address (8M)

- Setup C stack
- Initialize UART
- Decompress the kernel to 0x0
- Jump to the kernel entry

1.4 L1 cache invalidate hangs the CPU. WORKAROUND

When trying to enable the L1 cache in the following code the system hang:

```
/* Enable caches for 603's, 604, 750 & 7400 */
setup_common_caches:
mfspr r11,SPRN_HID0
andi. r0,r11,HID0_DCE
ori r11,r11,HID0_ICE|HID0_DCE
ori r8,r11,HID0_ICFI
bne 1f /* don't invalidate the D-cache */
ori r8,r8,HID0_DCI /* unless it wasn't enabled */
1: sync
mtspr SPRN_HID0,r8 /* enable and invalidate caches */
sync
mtspr SPRN_HID0,r11 /* enable caches */
sync
isync
blr
```

It's also been found that disabling entirely L1/L2/L3 doesn't help since the system hang when paging is turned on.

It's been noticed that MMU data address translation is turned on at Rommon level and it seems like Linux is not prepared for such a condition.

The **workaround** consist of enabling the L1 cache WITHOUT invalidating it:

```
/* Enable caches for 603's, 604, 750 & 7400 */

setup_common_caches:
mfspr r11,SPRN_HID0
ori r11,r11,HID0_ICE|HID0_DCE
sync
mtspr SPRN_HID0,r11 /* enable caches */
sync
isync
blr
```

1.5 Marvell Bridge initialization hangs. DONE

At boot time the call to “mv64360_disable_all_windows” hangs the system. This function does the following:

- Disable 32bit windows (don't disable cpu->mem windows)
- Disable 64bit windows
- Turn off PCI->MEM access cntl wins not in mv64360_64bit_windows[]
- Disable all PCI-><whatever> windows

After investigation it's been found that the system doesn't hang but instead loose the serial console. The **fix** consist of keeping the MV64x60_CPU2DEV_1_WIN window open since it correspond to DEV_1 CS associated with external Uart specific to hardware layout.

1.6 External Serial Port adjustment and Interruption Mapping. DONE

The external serial port is using a standard ns16550 compatible uart. The default serial port for the 74xx is usually part of the bridge thus to make use the external port a special BAT has to be setup to provide IO access:

```
/*Serial*/
mb();
mtspr(SPRN_DBAT2U, 0xf1501ffe);
mtspr(SPRN_DBAT2L, 0xf150002a);
mb();
```

The proper mapping of the external Serial port to the kernel is done by the following code:

```
#define PPC_SERIAL_0            0xF1500000
#define PPC_SERIAL_1            0xF1500010
/* Rate for the 1.8432 Mhz clock for the onboard serial chip */
#define UART_CLK                        1843200*2
#define BASE_BAUD                       ( UART_CLK / 16 )
#define STD_SERIAL_PORT_DFNS \
      { 0, BASE_BAUD, PPC_SERIAL_0, 77, STD_COM_FLAGS, /* ttyS0 */ \
                  io_type: SERIAL_IO_PORT, },                                   \
      { 0, BASE_BAUD, PPC_SERIAL_1, 93, STD_COM_FLAGS, /* ttyS1 */ \
                  io_type: SERIAL_IO_PORT, },
```

1.7 Watchdog deactivation. DONE

Rommon has the watchdog activated atstart up. Linux need to deactivated the watchdog since at this point in time there is no process or kernel thread that feed the dog. The Watchdog is part of the Marvell chipset and is supported by the kernel with minimal configuration change.

1.8 Get Memory size broken for 4GB of RAM. DONE

The following had to be done to provide support for 4GB of ram. 32Bit variable overflow and return 0 instead of 4Gb:

```
mv64x60_calc_mem_size(struct mv64x60_handle *bh,
    u32 mem_windows[MV64x60_CPU2MEM_WINDOWS][2])
```

```
 {
-       u32     i, total = 0;
+       u32     i;
+       u64     total=0;
        for (i=0; i<MV64x60_CPU2MEM_WINDOWS; i++)
                total += mem_windows[i][1];
-       return total;
+       return (u32)(min(total,(u64)0xffffffff));
 }
```

1.9 Networking. DONE

1.9.1 PHY address adjustment. DONE

From Rommon code inspection the PHY address for port 0,1,2 is 0,1,2 respectively.

1.9.2 Special PHY setting for hfr blade. DONE

From Rommon code inspection the Intel 971 PHY requires special setting to turn on the proper signal strength. Similar logic apply to Marvell PHY Alaska 88E1000.

Part of the PHY initialization code in Rommon has been ported to Linux. From this point on the link is negotiated as expected.

1.9.3 System hangs when configuring network. DONE

Tx and Rx buffers are managed via link list of descriptors. Network buffers are typically placed in DRAM and Tx/Rx descriptors in SRAM for performance reason. It's been found that when configuring the networks interface, the system hangs when the Tx/Rx descriptor are initialized.

The source of the problem is coming from the fact that the SRAM is NOT configured at system start-up OR the provided address is invalid.

For now the DRAM is going to be use to hold the Tx/Rx descriptor. Since DRAM is not cache coherent, the DMA allocation primitive provided in the kernel has to be used to maintain coherency.

The optimal configuration for the Rx/Tx descriptor is going to be the SRAM configured with the address decoding windows marked as a cache coherent region. Access to these regions results in a snoop on the CPU bus. This relief the CPU from doing the cache management for every packet in order to maintain coherency.

1.9.4 Marvell Rx/Tx DMA descriptor not working correctly. WORKAROUND

Network support for the MV64360 is already part of the MV64x60 Linux driver. It's been found that on the hfr-ppc 7455 blade the Tx/Rx descriptor are not functioning correctly. When transmitting data the

network controller is ending up into a weird state and finally give up on the transaction. The data buffers are stacking up on the stale TX queues.

The origin of the problem is around the address decoding logic. In fact in the actual configuration (4Gb of DRAM) the translation register logic is creating a shadow copy which in turn confuse the address decoding logic of the bridge when external device DMA into CPU memory.

The **workaround** consist of limiting the CPU2MEM windows to 2Gb of RAM.

Interesting enough this problem exist on the latest open source Linux kernel tree. It's more than likely than very few people are using 4Gb of RAM together with the Marvell controller!

1.9.5 Only 1 port active. DONE

The MV64360 has 3 integrated port. It's been observed that only port 0 is brought up. Port 0 correspond to INTEL 971 PHY where port 1 & port 2 correspond to MARVEL_88E1000 PHY.

The origin of the problem is around the "ethernet_phy_detect" function in the MV64x60 Linux driver. In fact this function should detect whether a PHY is present or not on a specified port by performing a simple read/modify/write/compare on a scratch pad register. This algo doesn't work for the Marvell PHY.

The **fix** consist of using part of the PHY detection code in Rommon. From this point on the PHY is detected has expected.

1.9.6 Active -> standby limited connectivity. DONE

Using port 0 the only communication channel that is operational is between active and standby
Active ping srp:

```
root:~> ping 12.29.19.103
PING 12.29.19.103 (12.29.19.103): 56 data bytes
64 bytes from 12.29.19.103: icmp_seq=0 ttl=64 time=57.7 ms
```

Srp ping Active:

```
root:~> ping 12.29.19.102
PING 12.29.19.102 (12.29.19.102): 56 data bytes
64 bytes from 12.29.19.102: icmp_seq=0 ttl=64 time=53.4 ms
```

From document *"HFR RPB: Hardware Functional Specification: EDCS459282, Rev. 2"* It's been found that the Intel LXT971A PHY is used to provide a FE loopback through the backplane. This PHY is connected to the port 0 . Port 1 is connected to the front panel and use the MARVEL_88E1000 PHY.

The **fix** consist of using port 1 when communicating with external devices.

1.10 PCI. DONE

PCI support is already part of the MV64x60 Linux driver. The interesting part of the work was to provide a PCI memory map compatible with the requirement found in the document *"CRS-MSC/B: Software Functional Spec:EDCS-488680,Rev.5"*

0x00000000 - 7FFFFFFF : (2Gig) physical memory
0x80000000 - BFFFFFFF : (1Gig) PCI0 memory space
0xC0000000 - C7FFFFFF : (128 MB) bootflash - lower bank
0xC8000000 - EFFFFFFF : (512 MB) Unused
0xF00000000 - 0xF00FFFFF : (1MB) device bus FPGA
0xF01000000 - 0xF01FFFFF : (1MB) Atlantis internal SRAM
0xF0200000 - 0xF02FFFFF : (1MB) PCI1 I/O space
0xF0400000 - 0xF047FFFF : (512kB) NVRAM
0xF0800000 - 0xF0FFFFFF : (8MB) BITS part
0xF1000000 - 0xF10FFFFF : (1MB) Atlantis internal registers
0xF1500000 - 0xF15FFFFF : (1MB) UART on the device bus
0xF1800000 - 0xF1BFFFFF : (4MB) PCI0 memory base addresses
0xF4000000 - 0xF43FFFFF : (4MB) PCI1 memory base addresses
0xF8000000 - 0xFFFFFFFF : (128MB) bootflash - upper bank

From Linux we observed that PCI MEM (second hose) is matching PCI1 memory base address and also that PCI I/O (second hose) is matching PCI1 I/O space. The pci name database has been added to the system in order to get human readable format:

root:~> lspci -v
00:01.0 PCMCIA bridge: Cirrus Logic CL 6729 (rev 07)
Flags: stepping, slow devsel
I/O ports at f0200000 [size=4]
00:02.0 IDE interface: Silicon Image, Inc. PCI0680 Ultra ATA-133 Host Controller (rev 02) (prog-if 85 [Master SecO PriO])
Subsystem: Silicon Image, Inc. PCI0680 Ultra ATA-133 Host Controller
Flags: bus master, medium devsel, latency 0
I/O ports at f0200100 [size=8]
I/O ports at f0200200 [size=4]
I/O ports at f0200300 [size=8]
I/O ports at f0200400 [size=4]
I/O ports at f0200500 [size=16]
Expansion ROM at <unassigned> [disabled] [size=512K]
Capabilities: [60] Power Management version 2
00:03.0 Network controller: Broadcom Corporation: Unknown device 5618 (rev 02)
Flags: bus master, 66Mhz, medium devsel, latency 0
Memory at 00000000f4000000 (64-bit, non-prefetchable) [size

1.11 Build environment. DONE

A new platform definition flag has been created to hold the specific change around Cisco HFR ppc-7455 in the Linux kernel build. The flag is CONFIG_CISCO_HFR_PPC.

The Linux kernel for the RP can now be build from any available Cisco Linux server example (sjc-xdm-002.cisco.com). The SRC as well as the tools chain can be found under "/ws/etmartin/project/xr_migration/".

The only limitation is that we cannot modify the Root File System layout since we don't have root access on those machine.

1.12 Disk driver and file system. DONE

The **IDE interface: Silicon Image, Inc. PCI0680 Ultra ATA-133 Host Controller** is mapped at PCI idsel 2 and the IRQ line is MPP 10 -> Linux IRQ 74.

The proper mapping is provided from the following function:

```
static int __init
hfr_irq_lookup(unsigned char idsel, unsigned char pin)
{
    static char pci_irq_table[][4] = {
        /*
         * PCI IDSEL/INTPIN->INTLINE
         *      A   B   C   D
         */
        { 0, 0, 0, 0 },
        { 74, 0, 0, 0 },       /*IDSEL 2 is IDE controller, Using IRQ A only*/
        { 0, 0, 0, 0 },
        { 0, 0, 0, 0 },
    };
    const long min_idsel = 1, max_idsel = 4, irqs_per_slot = 4;
    return PCI_IRQ_TABLE_LOOKUP;
```

The file system is mounted on a fat16 MSDOS partition.

2 Works that need to be done

2.1 Kernel support for 4Gb of RAM. TODO

The actual understanding of the memory map is:

0x00000000 - 0x7FFFFFFF : (2Gig) physical memory
0x80000000 - 0xBFFFFFFF : (1Gig) PCI0 memory space
0xC0000000 - 0xEFFFFFFF : (768MB) Unused
0xF00000000 - 0xF00FFFFF : (1MB) device bus FPGA
0xF01000000 - 0xF01FFFFF : (1MB) Atlantis internal SRAM

0xF0200000 - 0xF02FFFFF : (1MB) PCI1 I/O space
0xF0400000 - 0xF05FFFFF : (2MB) NVRAM
0xF0800000 - 0xF0FFFFFF : (8MB) BITS part
0xF1000000 - 0xF10FFFFF : (1MB) Atlantis internal registers
0xF1500000 - 0xF15FFFFF : (1MB) UART on the device bus
0xF1800000 - 0xF1BFFFFF : (4MB) PCI0 memory base addresses
0xF4000000 - 0xF43FFFFF : (4MB) PCI1 memory base addresses
0xFC000000 - 0xFFFFFFFF : (64MB) bootflash

Not sure to understand HOW 4GB of ram will get mapped?

2.2 Kernel version. TODO

The current kernel version is Linux 2.6.19.2 compiled using gcc tool chain version 4.0.0.

2.3 Root file system. TODO

The actual RFS is based on Busybox technologie..

3 Annex

3.1 Linux booting console output

Receiving /tftpboot/wsetmartin/zImage.elf from 223.255.254.254 !!
tftp_process_packet: last packet, block=5160, size=450.File reception completed.
Headersize=2641858, oursize=2641858 .... Linux on Cisco's box
CPU PVR 80020102
Total Ram 4095 Meg
mv64x60 bridge base addr 0100F100
mv64x60 CPU Bar enable 00073200
mv64x60 bank0 base addr 00000000
mv64x60 bank0 size 00003FFF
mv64x60 bank1 base addr 00004000
mv64x60 bank1 size 00003FFF
mv64x60 bank2 base addr 00010000
mv64x60 bank2 size 00003FFF
mv64x60 bank3 base addr 00014000
loaded at:00800000 00A76134
zimage at: 00804F25 008FA8C1
initrd at: 008FB000 00A73402
avail ram: 00400000 00800000 Linux/PPC load: console=ttyS0,9600 root=/dev/ram
Uncompressing Linux...done.
Now booting the kernel
Early serial text: Entering Linux land
id mach(): done
MMU:enter
MMU:hw init
hash:enter
hash:find piece
hash:patch
hash:done
MMU:mapin
MMU:setio
MMU:exit
setup_arch: enter
setup_arch: bootmem
ev64360_setup_arch: enter
ev64360_setup_arch: calling setup_bridge
mv64x60 initialization
ev64360_setup_arch: exit
arch: exit
mv64360_init_irq: enter
mv64360_init_irq: exit
Total memory = 768MB; using 2048kB for hash table (at c0400000)
Linux version 2.6.19.2 (etiennem@work) (gcc version 4.0.0 (DENX ELDK 4.1 4.0.0)) #177 PREEMPT Thu May 8

12:23:17 EDT 2008
mv64x60_wdt_service
Cisco Systems hfr-ppc7455 port (C) 2008 Etienne Martineau(etmartin@cisco.com)
Zone PFN ranges:
 DMA 0 -> 196608
 Normal 196608 -> 196608
early_node_map[1] active PFN ranges
 0:0 -> 196608
Built 1 zonelists. Total pages: 195072
Kernel command line: console=ttyS0,9600 root=/dev/ram
PID hash table entries: 4096 (order: 12, 16384 bytes)
time_init: decrementer frequency = 33.333333 MHz
mv64x60_wdt_service
Console: colour dummy device 80x25
Dentry cache hash table entries: 131072 (order: 7, 524288 bytes)
Inode-cache hash table entries: 65536 (order: 6, 262144 bytes)
Memory: 773120k available (1640k kernel code, 476k data, 136k init, 0k highmem)
Mount-cache hash table entries: 512
checking if image is initramfs...it isn't (no cpio magic); looks like an initrd
Freeing initrd memory: 1505k freed
NET: Registered protocol family 16
PCI: Probing PCI hardware
NET: Registered protocol family 2
IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
TCP established hash table entries: 131072 (order: 7, 524288 bytes)
TCP bind hash table entries: 65536 (order: 6, 262144 bytes)
TCP: Hash tables configured (established 131072 bind 65536)
TCP reno registered
Thermal assist unit not available
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
Generic RTC Driver v1.07
MV64x60 watchdog driver
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing disabled
serial8250: ttyS0 at I/O 0xf1500000 (irq = 77) is a 16550A
serial8250: ttyS1 at I/O 0xf1500010 (irq = 93) is a 16550A
RAMDISK driver initialized: 16 RAM disks of 32768K size 1024 blocksize
loop: loaded (max 8 devices)
MV-643xx 10/100/1000 Ethernet Driver
eth0: port 0 with MAC address 00:00:00:00:00:00
eth0: Scatter Gather Enabled
eth0: TX TCP/IP Checksumming Supported
eth0: RX TCP/UDP Checksum Offload ON
eth0: RX NAPI Enabled
mice: PS/2 mouse device common for all mice
TCP cubic registered

NET: Registered protocol family 1
NET: Registered protocol family 17
RAMDISK: Compressed image found at block 0
EXT2-fs warning: feature flags set on rev 0 fs, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem) readonly.
Freeing unused kernel memory: 136k init
mv64x60_wdt: watchdog deactivated
root:~> ### Application running ...

3.2 Linux Kernel object section

vmlinux: file format elf32-powerpc

vmlinux

architecture: powerpc:common, flags 0x00000112:

EXEC_P, HAS_SYMS, D_PAGED

start address 0xc0000000

Program Header:

LOAD off 0x00010000 vaddr 0xc0000000 paddr 0xc0000000 align 2**16

filesz 0x00247086 memsz 0x0025d708 flags rwx

Sections:

Idx Name Size VMA LMA File off Algn

0 .text 001ba7f8 c0000000 c0000000 00010000 2**5

CONTENTS, ALLOC, LOAD, READONLY, CODE

1 .rodata 00021c3d c01bb000 c01bb000 001cb000 2**3

CONTENTS, ALLOC, LOAD, READONLY, DATA

2 .pci_fixup 00000460 c01dcc40 c01dcc40 001ecc40 2**2

CONTENTS, ALLOC, LOAD, READONLY, DATA

3 .rio_route 00000000 c01dd0a0 c01dd0a0 00257086 2**0

CONTENTS

4 __ksymtab 00000000 c01dd0a0 c01dd0a0 00257086 2**0

CONTENTS

5 __ksymtab_gpl 00000000 c01dd0a0 c01dd0a0 00257086 2**0

CONTENTS

6 __ksymtab_unused 00000000 c01dd0a0 c01dd0a0 00257086 2**0

CONTENTS

7 __ksymtab_unused_gpl 00000000 c01dd0a0 c01dd0a0 00257086 2**0

CONTENTS

```
 8 __ksymtab_gpl_future 00000000  c01dd0a0  c01dd0a0  00257086  2**0
                CONTENTS
 9 __kcrctab     00000000  c01dd0a0  c01dd0a0  00257086  2**0
                CONTENTS
10 __kcrctab_gpl 00000000  c01dd0a0  c01dd0a0  00257086  2**0
                CONTENTS
11 __kcrctab_unused 00000000  c01dd0a0  c01dd0a0  00257086  2**0
                CONTENTS
12 __kcrctab_unused_gpl 00000000  c01dd0a0  c01dd0a0  00257086  2**0
                CONTENTS
13 __kcrctab_gpl_future 00000000  c01dd0a0  c01dd0a0  00257086  2**0
                CONTENTS
14 __param       0000021c  c01dd0a0  c01dd0a0  001ed0a0  2**2
                CONTENTS, ALLOC, LOAD, READONLY, DATA
15 __ex_table    00001470  c01de000  c01de000  001ee000  2**3
                CONTENTS, ALLOC, LOAD, READONLY, DATA
16 __bug_table   00004740  c01df470  c01df470  001ef470  2**0
                CONTENTS, ALLOC, LOAD, READONLY, DATA
17 .data         000363f0  c01e4000  c01e4000  001f4000  2**5
                CONTENTS, ALLOC, LOAD, DATA
18 .data.page_aligned 00003000  c021b000  c021b000  0022b000  2**12
                CONTENTS, ALLOC, LOAD, DATA
19 .data.cacheline_aligned 000040c0  c021e000  c021e000  0022e000  2**5
                CONTENTS, ALLOC, LOAD, DATA
20 .data.init_task 00002000  c0224000  c0224000  00234000  2**2
                CONTENTS, ALLOC, LOAD, DATA
21 .init.text    0001bacc  c0226000  c0226000  00236000  2**2
                CONTENTS, ALLOC, LOAD, READONLY, CODE
22 .exit.text    00000a64  c0241acc  c0241acc  00251acc  2**2
                CONTENTS, ALLOC, LOAD, READONLY, CODE
23 .init.data    0000357c  c0242530  c0242530  00252530  2**3
                CONTENTS, ALLOC, LOAD, DATA
24 .init.setup   00000258  c0245ab0  c0245ab0  00255ab0  2**2
                CONTENTS, ALLOC, LOAD, DATA
```

25 .initcall.init 000001b8 c0245d08 c0245d08 00255d08 2**2

CONTENTS, ALLOC, LOAD, DATA

26 .con_initcall.init 00000004 c0245ec0 c0245ec0 00255ec0 2**2

CONTENTS, ALLOC, LOAD, DATA

27 .security_initcall.init 00000000 c0245ec4 c0245ec4 00257086 2**0

CONTENTS

28 __ftr_fixup 00000240 c0245ec4 c0245ec4 00255ec4 2**2

CONTENTS, ALLOC, LOAD, READONLY, DATA

29 .init.ramfs 00000086 c0247000 c0247000 00257000 2**0

CONTENTS, ALLOC, LOAD, READONLY, DATA

30 .bss 00015708 c0248000 c0248000 00257086 2**5

ALLOC

31 .stab 00000528 00000000 00000000 00257088 2**2

CONTENTS, READONLY, DEBUGGING

32 .stabstr 00000753 00000000 00000000 002575b0 2**0

CONTENTS, READONLY, DEBUGGING

33 .comment 00004da8 00000000 00000000 00257d03 2**0

CONTENTS, READONLY

34 .note.GNU-stack 00000000 00000000 00000000 0025caab 2**0

CONTENTS, READONLY, CODE

3.3 Linux wrapper object section

architecture: powerpc:common, flags 0x00000112:

EXEC_P, HAS_SYMS, D_PAGED

start address 0x00800000

Program Header:

LOAD off 0x00010000 vaddr 0x00800000 paddr 0x00800000 align 2**16

filesz 0x0010d000 memsz 0x0010f134 flags rwx

STACK off 0x00000000 vaddr 0x00000000 paddr 0x00000000 align 2**2

filesz 0x00000000 memsz 0x00000000 flags rwx

Sections:

Idx Name Size VMA LMA File off Algn

0 .text 00003af0 00800000 00800000 00010000 2**2

CONTENTS, ALLOC, LOAD, READONLY, CODE

```
1 .data         00109000  00804000  00804000  00014000  2**2
                CONTENTS, ALLOC, LOAD, DATA
2 .bss          00002134  0090d000  0090d000  0011d000  2**2
                ALLOC
3 .note.GNU-stack 00000000  00000000  00000000  0011d000  2**0
                CONTENTS, READONLY
```